Job Control Guide

Schrödinger Suite 2006

Schrödinger Press

Revision A, April 2006

Contents

Document Conventions

In addition to the use of italics for names of documents, the font conventions that are used in this document are summarized in the table below.

Table 1.1.

Font	Example	Use
Sans serif	Project Table	Names of GUI features, such as panels, menus, menu items, buttons, and labels
Monospace	`$SCHRODINGER/maestro`	File names, directory names, commands, environment variables, and screen output
Italic	*filename*	Text that the user must replace with a value
Sans serif uppercase	CTRL+H	Keyboard keys

In descriptions of command syntax, the following UNIX conventions are used: braces `{ }` enclose a choice of required items, square brackets `[ ]` enclose optional items, and the bar symbol | separates items in a list from which one item must be chosen. Lines of command syntax that wrap should be interpreted as a single command.

In this document, to *type* text means to type the required text in the specified location, and to *enter* text means to type the required text, then press the ENTER key.

References to literature sources are given in square brackets, like this: [10].

Introduction

Nearly all computational jobs launched from Maestro are run under Schrödinger's Job Control facility. The Job Control facility provides a uniform mechanism for launching, monitoring and controlling calculations, both for jobs launched from Maestro and for jobs launched from the command line. The Job Control facility keeps information on jobs in a database that is set up for each user.

The provision of a Job Control facility makes it easy to run Schrödinger software. You can submit jobs *from* any computer *to* any computer, cluster, or grid, or to a batch queue, and have the results returned to the computer from which you submitted the job. The process is independent of the particular platform from which you submit the job or on which you run the job. The platform details are hidden, but that also means that the details have to be communicated to Job Control.

This manual provides information on all aspects of Job Control. Chapter 2 describes the basics of setting up your computers for Job Control and setting up the information on these computers that is needed by Job Control. Chapter 3 describes how to set up the information that Job Control needs to interact with batch queues. Chapter 4 describes the life cycle of a job and provides information on submitting jobs and how the job execution is set up. Chapter 5 provides information on the job database and managing jobs.

Preparing for Job Submission

The default installation of Schrödinger software allows you to run jobs locally on a single processor, without doing any preparation. The only requirement is that you set the environment variable `SCHRODINGER` to the installation directory.

If you want to submit jobs to other computers (or *hosts*) or submit distributed or parallel jobs, you must configure the available hosts to allow job submission from another host and provide the needed information to Job Control. This chapter describes how to configure hosts and to provide the information that is needed to run jobs directly on remote hosts. The extra information that is needed to submit jobs to batch queues is described in Chapter 3.

Information on preparing for job submission is also contained in the *Installation Guide*, including preparing for submission of jobs to batch queues and grid servers.

2.1 The Hosts File

The Job Control facility obtains information about the computers (hosts) on which it will run jobs from the *hosts file*. The default name for this file is `schrodinger.hosts`. Maestro also uses the hosts file to set up the menus in the Start dialog box.

The default installation of the hosts file allows you to run single-processor jobs on the local host, and no further action is required. However, if you wish to run distributed jobs, run jobs remotely, use batch queues, change the location of the scratch directory, or set up the remote environment, then you must specify the required information in this file.

The hosts file consists of one or more *entries*, each of which describes a configuration for running jobs on a given host. Each entry consists of a number of settings, one per line. For each entry in the `schrodinger.hosts` file, the following basic settings can be made:

`name:` *entry-label*
`host:` *hostname*
`user:` *username*
`tmpdir:` *tmpdir*
`processors:` *number of processors*
`schrodinger:` *installation-path*
`env:` *environment variable assignment*

The settings can be formatted with any combination of spaces and tabs, but the entire setting must be on one line. The setting keywords are not case-sensitive. Comments can be included by beginning a line with a # sign.

The settings are listed with a concise description in Table 2.1. The basic settings are described in the following sections. If the host has a batch queueing system, you can add settings for the batch queue—see Chapter 3 for more information. For grid computing there are special settings for the supported architectures—see for more information.

Table 2.1. Keywords for `schrodinger.hosts` *file settings.*

Keyword	Description
`name`	The name of the entry. For a host this is usually the host name, but any name can be used. This name is displayed in Maestro by job control, and is used for the `-HOST` option. The value `localhost` is a special value that means the host on which the job is launched.
`host`	The host name. Required for batch queue entries; otherwise only needed if it is different from `name`.
`schrodinger`	The path to the Schrödinger software installation on the host. You can include more than one of these settings.
`user`	The user name to use on the host. This should never be set in the hosts file in the installation directory. It is required if the user has a different user name on the defined host than on the host on which the job is launched.
`processors`	The number of processors available on the host. If the host is part of a cluster, this number should be the total number of processors available on the cluster. The default is 1.
`env`	Environment variables to be set on the specified host. The syntax for the environment variables is *variable=value*, regardless of the shell used. List each environment variable on a separate `env` line. Not used on the submission host.
`tmpdir`	Base directory for temporary or scratch files, also called the scratch directory. The actual directory created for scratch files (the job directory) is *tmpdir*`/`*username*`/`*uniquename*, where *tmpdir* is the directory defined here and *username* is the user name. Multiple `tmpdir` settings can be added for a given host and are used by Maestro, but the first setting is used otherwise.
`queue`	Queuing system name. PBS, SGE and LSF are the three supported systems. Must be set to the subdirectory of `$SCHRODINGER/queues` that contains the support files for the queuing system.
`qargs`	Arguments to be used when submitting jobs to a batch queue. These arguments should specify any parameters that define the queue.

You can create multiple entries for a given host with different settings. For example, you might want to create two entries for a host with different `tmpdir` settings to make use of different default temporary storage areas.

A sample `schrodinger.hosts` file is shown below.

```
# Schrodinger hosts file
#
name:        localhost
schrodinger: /software/schrodinger
#
name:        ahost
#
name:        bhost
#
name:        old_bhost
host:        bhost
schrodinger: /software/schrodinger_old
#
name:        another_host
processors:  2
tmpdir:      /scr
schrodinger: /usr/bin/share/schrodinger
#
# End of Schrodinger hosts file
```

2.1.1 The name and host Settings

The `name` setting must be the first line for each entry. This is the name that is used to select the host (or batch queue) with the configuration specified in the following settings. It is displayed in the list of hosts in the Start dialog box Host menu. Usually, *entry-label* is the name of a host that can be used to run a calculation. If it is not, you must include a `host` setting that supplies the host name. The `host` setting is only needed if the name line does not give the host address. You might, for example, want to provide an alias in the `name` setting and define the host name in a `host` setting if the host name is long. Another use of multiple entries for a single host is to specify different settings on a host, such as different scratch directories or different software installations. You can also use the `name` and `host` settings to specify a batch queue name and the host on which the batch system is available.

The host name does not need to be the fully qualified domain name unless the host on which you plan to run (the *execution* host) is not on the same local network as the host from which you plan to submit jobs (the *submission* host).

The value localhost is a special name setting that means the host from which the job was submitted. In addition to this function, the settings for the localhost entry are used as the default values for all other entries. In the schrodinger.hosts file example above, the host entries ahost and bhost inherit the schrodinger setting from the localhost entry.

If you run jobs from the command line, the name setting is what you should use with the -HOST option to select the hosts to run the job.

2.1.2 The user Setting

If you have different user names on the submission and execution hosts, you must include a user setting for the execution host in the hosts file on the submission host. The user setting should never be added to entries in the hosts file in the installation directory, because this would prevent other users from using those entries. If a user setting is required, the hosts file should be copied by the user to the directory $HOME/.schrodinger on the submission hosts and the user settings added to this copy.

2.1.3 The tmpdir Setting

The tmpdir setting specifies the scratch directory, where temporary files can be written. Examples are /scr or /temp. The file system on which this directory is mounted should be large enough for the largest temporary files, should be mounted locally, and should be writable by the user. Do not use symbolic links, as these can cause some programs to fail. The actual directory created for scratch files (the job directory) is *tmpdir*/*username*/*uniquename*, where *tmpdir* is the directory defined here and *username* is the user name.

You can include multiple tmpdir settings for a given host. These settings are listed by Maestro in the Scratch directory option menu of the Start dialog box, and can be selected for a job. If you do not start a job from Maestro, the first tmpdir setting is used and the others are ignored.

If you do not specify tmpdir for a host, the tmpdir setting from the localhost entry is used, if there is one. Otherwise, the scratch directory is set to $HOME/.schrodinger/tmp. The use of the home file system for large temporary files is discouraged in most places, so you should always ensure that tmpdir is defined for the hosts you run jobs on, if the job requires temporary storage.

You can override the tmpdir setting in the schrodinger.hosts file by setting the SCHRODINGER_TMPDIR environment variable or using the -TMPDIR command-line option—see Section 4.3 on page 26 for more information. For example, if the directory designated by tmpdir becomes full with files that you don't have permission to delete, you can set SCHRODINGER_TMPDIR to a different directory and continue to run jobs.

2.1.4 The processors Setting

For stand-alone computers with multiple processors, set `processors` to the number of processors in the computer. For batch queues on computer clusters, set `processors` for each cluster node to the *total* number of processors in the entire cluster.

2.1.5 The schrodinger Setting

The `schrodinger` setting specifies the directory in which your Schrödinger software is installed on this host (the *installation* directory). This setting allows you to choose between software versions to use on the remote machine, if more than one version is installed. You can provide multiple `schrodinger` settings for a single host entry. You may want to do this if you have installations of several versions of Schrödinger software in different directories.

For example, suppose your Schrödinger software was installed in `/usr/bin/schrodinger`, and you have a cluster in which the software is installed in `/storage/schrodinger`. If the cluster is named `mycluster`, and `/storage` is only accessible to the cluster, you could set up your hosts file with the following schrodinger settings:

```
name:        localhost
schrodinger: /usr/bin/schrodinger

name:        clus4hr
host:        mycluster
queue:       PBS
qargs:       -l walltime=04:00:00
schrodinger: /storage/schrodinger
```

In this example, the cluster is running queuing software. Another way of making the settings is to include both `schrodinger` settings for the `localhost` entry:

```
name:        localhost
schrodinger: /usr/bin/schrodinger
schrodinger: /storage/schrodinger

name:        clus4hr
host:        mycluster
queue:       PBS
qargs:       -l walltime=04:00:00
```

In this case, both `schrodinger` settings are used in the default search for software on any host, not just on `mycluster`.

You can override the `schrodinger` setting with the `-VER` and `-REL` options if you run a program from the command line. See Table 4.4 on page 28 for details.

2.1.6 The env Setting

The `env` setting specifies an environment variable that is to be set on this host when any job is started. The syntax of the setting is *variable=value* (regardless of the UNIX shell used), where *variable* is the environment variable and *value* is its value. For example,

```
env: SCHRODINGER_THIRDPARTY=/software/databases
```

To set multiple environment variables, include one `env` setting for each variable. Environment variables set in the hosts file take precedence over any that are set in your UNIX shell, either on the local host or the remote host.

2.1.7 Customizing the Hosts File

You can copy and edit the `schrodinger.hosts` file from the installation directory (`$SCHRODINGER`) to customize its settings. You usually do not need to do this unless you have different user names on different hosts. If you have installed Schrödinger products on multiple hosts, you may need to edit the `schrodinger.hosts` file on each host to add entries for the other hosts.

2.1.8 Testing the Hosts File

The `schrodinger.hosts` file can be tested by running the following command in the directory containing the file:

```
$SCHRODINGER/hunt -rtest
```

This command attempts to contact each of the hosts listed, and thus serves as a check on host accessibility and on whether remote access has been set up appropriately on the various hosts.

2.2 Setting Up Access to Remote Hosts

To run remote or distributed jobs, users must be able to execute commands on another computer or another node of a multiprocessor computer or cluster by using the `rsh` command or the `ssh` command without specifying a password. The `rsh` command is the default command used by Job Control, but you can also use `ssh`, which is more secure and is recommended on clusters. Setting up access using `rsh` is described in the next subsection, and setting up access using `ssh` is described in the following subsection.

Note: To be able to run jobs on remote hosts, Schrödinger products must be installed on both the submission host and the execution hosts, or on a file system that is accessible to both.

If you have a `schrodinger.hosts` file, you can automatically check the connections to all the hosts listed in it using the command

```
$SCHRODINGER/hunt -rtest
```

See Section 2.1.8 on page 8 for more information about this command.

2.2.1 Setting Up Access with rsh

To set up access to remote hosts using `rsh` without passwords, you must do one of the following:

- Create or modify the `hosts.equiv` file in the `/etc` directory on each host. This file should contain a list of hosts from which users can log in without giving their passwords (provided that their userids are the same on each of the hosts). Creating a `hosts.equiv` file usually requires root permission.
- Create or modify the `.rhosts` file in the user's home directory on each host. The `.rhosts` file should list the names of the hosts and the user name from which the user logs in without specifying a password. The list should contain two lines for each machine—one with the host name alone and one with the fully qualified name, as follows:

  host username
  host`.`*domain username*

  The *username* in the `.rhosts` file is optional if the user name is the same on the remote hosts as on the submission host.

  You do not need root permission to configure this file, but you must make sure that the file does not have "group" or "other" write permission. To ensure the correct permissions, use one of the following commands

  ```
  chmod 644 $HOME/.rhosts
  chmod go-w $HOME/.rhosts
  ```

  You might also want to remove read permissions for "group" and "other".

If you intend to run jobs on a cluster, you must include each node in the cluster in the list of hosts in the `hosts.equiv` file or the `.rhosts` file.

Contact your system administrator or consult your queuing system and cluster documentation in case there are special requirements for the `hosts.equiv` or `.rhosts` files for your particular queuing system or cluster setup.

Once you have set up the `hosts.equiv` file or the `.rhosts` file, use the following command to check for successful communication between the host that the job will be started on and each of the other hosts that the job will use.

```
rsh [-l username] hostname date
```

This command should print the date from the host *hostname*.

2.2.2 Setting Up Access with ssh

To use passwordless `ssh`, the hosts to which you want to connect must be configured to satisfy the following requirements:

- An `sshd` server must be running.
- RSA public key authentication must be enabled and empty passphrases must be allowed in the `sshd` configuration.

  Note: Public key authentication is enabled in OpenSSH by default.

The following steps allow you to use `ssh` between computers that share your login directory without specifying a password.

1. Generate your public/private RSA key pair:

   ```
   cd ~/.ssh
   ssh-keygen -t rsa
   ```

   Note: When asked for a passphrase *do not* enter one; just press ENTER. If you specify a passphrase it defeats the purpose of configuring passwordless `ssh`.

2. Add your public key to the list of keys allowed to log in to your account:

   ```
   cat id_rsa.pub >> authorized_keys
   cat id_rsa.pub >> authorized_keys2
   ```

   The two separate files are necessary to support both OpenSSH 1.5 and OpenSSH 2.0 protocols. Some versions use just one or the other of these files.

3. Suppress the confirmation dialog you ordinarily get when you connect to a machine for the first time:

   ```
   echo "StrictHostKeyChecking no" >> config
   ```

   This is necessary if you want to use `ssh` non-interactively and you cannot get RSA signatures for every host to which you want to allow connections in your `known_hosts` file ahead of time.

4. Remove your known_hosts file:

   ```
   rm known_hosts*
   ```

   This is necessary so that the new RSA key-pair mechanism is used for every host. Otherwise, hosts to which you previously connected using passwords might not use the new system automatically.

5. Make sure your home directory cannot be written by anyone but you:

   ```
   chmod go-w ~
   ```

   This is required before ssh will allow passwordless access to your account.

6. To make sure that Job Control uses ssh instead of rsh, set the environment variable SCHRODINGER_RSH to ssh:

   csh, tcsh: `setenv SCHRODINGER_RSH ssh`

   bash, ksh: `export SCHRODINGER_RSH=ssh`

   This only needs to be done on the machines where you want to use ssh instead of rsh. For instance, you might want to continue using rsh on your LAN and force ssh to be used only on your cluster. The easiest way to force ssh to be used only on certain hosts is to add the following line to the entries for those hosts in your schrodinger.hosts file:

   ```
   env:  SCHRODINGER_RSH=ssh
   ```

Use of ssh on clusters is recommended, since rsh has access to a limited number of ports and is more likely to result in job failure as a consequence.

2.3 Environment Variables

There are several environment variables that can be used to specify the location of resources, manage the job database, and control various aspects of job execution. These environment variables are given in Table 2.2. Some of these environment variables provide alternate means of specifying the resource; others are the sole means of specifying a resource or of overriding a default value.

Table 2.2. Environment variables for resource specification and job control.

Variable	Purpose
SCHROD_LICENSE_FILE LM_LICENSE_FILE	Specify the license file to use if the license file is not installed in $SCHRODINGER. The SCHROD_LICENSE_FILE definition supersedes any other definition. Both of these environment variables are FlexLM environment variables. Use SCHROD_LICENSE_FILE when LM_LICENSE_FILE does not point to a license file containing Schrödinger licenses. See Installing the License in the *Installation Guide*.
SCHRODINGER	Specify the installation directory.
SCHRODINGER_HOSTS	Specify the hosts file. See Section 4.3.4 on page 29.
SCHRODINGER_LICENSE_RETRY	Specify the maximum time to keep trying to obtain a license. The value can be an integer value in seconds, or an integer value with a time unit, such as 7200s, 120m, 2h. Default: 10 minutes.
SCHRODINGER_JOB_DEBUG	Control debugging output. Allowed values are: 0 No debugging output 1 Standard debugging output (same as entering -DEBUG from the command line) 2 Detailed debugging output (same as entering -DDEBUG from the command line)
SCHRODINGER_JOB_DISPOSITION	Specify how to incorporate job results into the project (same as entering -DISP from the command line) Available values are: append add entries to project replace replace project entries with new entries ignore do not incorporate entries Default: ignore See Section 4.8 on page 33.
SCHRODINGER_JOBDB	Specify the full path name for the job database Default: ~/.schrodinger/.jobdb See Section 5.1 on page 35.
SCHRODINGER_JOBDB_CLEANUP	Specify how long the completed job record is kept in the job database before being automatically deleted. For example, one day can be expressed as 86400, 1440m, 24h, or 1d. Default: 7 days. See Section 5.3.3 on page 42.
SCHRODINGER_PROJECT	Specify the path name of the Maestro project into which the results of the job should be incorporated (same as entering -PROJ from the command line) See Section 4.8 on page 33.

Table 2.2. Environment variables for resource specification and job control. (Continued)

Variable	Purpose
SCHRODINGER_RSH	Specify the command to use for remote connection (Section 2.2.2). You can use `rsh` or `ssh`, if they are specified in your path, otherwise you must use the full path name to the command.
SCHRODINGER_TMPDIR	Specify the full path name for the directory in which runtime temporary job directories are created. Overrides the `tmpdir` setting specified in the `schrodinger.hosts` file (Section 2.1.3).

Setting Up for Batch Queues and Grid Computing

Job Control provides basic support for submitting jobs to batch queues. Schrödinger currently supplies support for the PBS, SGE and LSF queueing systems in the standard software installation. Enabling batch queue submissions to a supported queueing system usually only requires the addition of a few lines to the `schrodinger.hosts` file and the specification of the queueing system and the queue name. These additions are described in the next section. For SGE installations you might also have to edit the `config` file to ensure that the environment for the queuing software is set up properly.

It should be reasonably straightforward to configure a Schrödinger software installation to support other queueing systems as well. The components required to support a batch system are a few text files that can be added or modified after installation. The nature of these files is explained in the following section.

3.1 Adding Batch Queue Support to the Hosts File

To enable job submissions to a batch queue on any queueing system, you must add host entries that define the available queues to the hosts file, `schrodinger.hosts`. There are two settings that define the queue: the `Queue` setting and the `Qargs` setting. These settings are described briefly in Table 2.1 on page 4. A sample of the host entries to be inserted into the hosts file is shown below:

```
# Batch submission to 'bigjobs' queue under PBS
Name: bigq
Host: cluster
Queue: PBS
Qargs: -q bigjobs
tmpdir: /storage/TMPDIR
#
# Batch submission to 'shortjobs' queue under PBS
Name: shortq
Host: cluster
Queue: PBS
Qargs: -q shortjobs
tmpdir: /storage/TMPDIR
```

This example defines two entries named `bigq` and `shortq` to which jobs can be sent on the host `cluster`.

The Job Control facility distinguishes batch queues from hosts by the presence of the `Queue` setting, which specifies the queueing system. The `Queue` setting must be set to the subdirectory of `$SCHRODINGER/queues` that contains the support files for the queueing system. The subdirectories for the supported queueing systems are `SGE`, `PBS`, and `LSF`. The `Qargs` setting specifies command line arguments for the queueing system's job submission command; for SGE, for instance, this is the `qsub` command.

You must also include a `host` setting because the `name` setting is used to specify the queue. Like normal remote host entries, host entries for batch queues inherit settings made in the `localhost` entry of the `schrodinger.hosts` file. If the queuing software is available to all hosts to which you have access, you should set `host` to `localhost`.

Batch queue entries can also have any of the other settings that host entries have, such as `schrodinger` and `tmpdir`. For queues on clusters, the `tmpdir` setting is required and should refer to a directory that is available to all the nodes and writable by all users who will use that queue. On shared memory machines, the `tmpdir` setting is optional.

You should also consider adding an `env` setting to set the `SCHRODINGER_LICENSE_RETRY` environment variable, particularly if there is likely to be a communication delay in obtaining a license or if the license pool is oversubscribed. See Section 2.3 on page 11 for the syntax.

3.2 Configuring Support for a Queueing System

To allow job submission to a batch queueing system, Job Control requires the following text files to be installed on the submission host and on any host that will run the queue submission command:

1. A `submit` script, which is a wrapper for the queueing system's own job submission utility (`qsub` for PBS and SGE, and `bsub` for LSF).
2. A `cancel` script, which is a wrapper for the queueing system's job removal command (`qdel` for PBS and SGE, and `bkill` for LSF).
3. A `config` file, which contains settings for the keywords `QPATH`, `QSUB`, `QDEL`, `QSTAT`, and `QPROFILE`. For the supported queuing systems, this is the only file that you should have to change, because it contains the path to the queuing software. The default submit and cancel scripts are defined in terms of these settings.

   As an example, `$SCHRODINGER/queues/PBS/config` contains the settings:

   ```
   QPATH=/usr/local/pbs/bin
   QSUB=qsub
   QDEL=qdel
   QSTAT=qstat
   ```

The `QPROFILE` keyword specifies the absolute path on the queue host of a configuration file that needs to be sourced to set up the environment to use the queue. This variable is useful for setting up an environment for the queuing system that does not affect the global environment. This setting is only supported for SGE, where it may be needed.

4. A `template.sh` file, which is a template for the shell script that is actually submitted to the batch queue and used to launch your calculation on the execution host.

These files are installed in a subdirectory of the `$SCHRODINGER/queues` directory, and must be installed in the *same* path on the submission host and the *queue host* (the host that runs the queueing software).

The name of this subdirectory is used as the name of the queueing system for the purposes of the hosts file, as described above. The standard software installation creates `$SCHRODINGER/queues/PBS`, `$SCHRODINGER/queues/SGE`, and `$SCHRODINGER/queues/LSF` directories, containing `submit`, `cancel`, `config`, and `template.sh` files for the PBS, SGE and LSF systems.

To modify these files or to provide new ones for an unsupported queueing system, it is necessary to understand what the job control system requires from each one. Each of the scripts is discussed below.

3.2.1 The submit Script

The `submit` script needs to support the command line syntax:

`submit` *job-script* [*qsub-options*]

where *job-script* is the name of a shell script that starts a job on the queue. This is always the first (and possibly only) command line argument to `submit`. Anything else on the command line must be passed on as arguments to the actual job-submission command.

If job submission is successful, `submit` should extract the batch ID from the output of the underlying job-submission command and report it in its output, in the form:

`BatchId:` *batchid*

If job submission fails for some reason, the script should exit with a non-zero exit code.

If you are creating your own `submit` script to support a new queueing system, you can use the `submit` scripts provided for PBS, SGE and LSF as templates. Use the `QSUB` variable rather than the actual submission command in your script, and define `QSUB` in the `config` file.

3.2.2 The cancel Script

The `cancel` script must support the command line syntax:

`cancel` *batchid*

where *batchid* is a batch ID assigned by the queueing system. Job Control keeps track of the batch ID of each submitted job so that the ID can be used for cancelling jobs. The `cancel` script should probably also return a nonzero exit status if the operation fails, but at present, Job Control ignores the exit status.

If you are creating your own `cancel` script to support a new queueing system, you can use the `cancel` scripts provided for PBS, SGE and LSF as templates. Use the `QSUB` variable rather than the actual submission command in your script, and define `QSUB` in the config file.

3.2.3 The Job Script Template File

The `template.sh` file is a skeleton for the Bourne-shell script that is actually submitted to the batch queue. Job Control takes this file and inserts the commands necessary to launch the particular job, and then submits the resulting file to the queueing system using the `submit` command described above. You might want to customize this file, for example to add email notification.

The following information from the `template.sh` file supplied for the LSF system illustrates how the `template.sh` file works.

```
#!/bin/sh
#BSUB -J %NAME%
#BSUB -o %DIR%/%JOBID%.qlog

export SCHRODINGER_BATCHID
SCHRODINGER_BATCHID=$LSB_JOBID

%ENVIRONMENT%

%COMMAND%
```

The `#BSUB` lines are directives that are interpreted by LSF. In this case, the first directive sets the LSF job name for this job to the Schrödinger job name, while the second specifies that any output from the job submission script should go to the file *jobid*`.qlog` in the directory from which the job was launched. Most other queueing systems also allow directives to be provided in the initial comment lines of the job submission scripts.

Table 3.1. Batch script variables.

Variable	Function
%NAME%	Schrödinger job name, usually derived from your input file name.
%DIR%	Directory from which the job was submitted.
%HOST%	Machine from which the job was submitted.
%USER%	User name of the user who submitted the job.
%JOBID%	Job ID assigned by the Job Control facility.
%ENVIRONMENT%	Commands that define environment variables that are required for the job to run.
%JOBDB%	The path to the job database.
%NPROC%	Number of processors that were requested.
%LOGDIR%	The directory in which log files are written.
%HOME%	Home directory on the submission host.
%COMMAND%	Command that starts the `jmonitor` program, which sets up, runs, and cleans up after your calculation. Must be the last command in the script.

The words delimited by percent signs are variables, which are replaced at job launch time with the actual job name, Schrödinger job ID, and so on, for the job you are submitting. Variables that you can put in any new `template.sh` file are listed in Table 3.1.

The `%ENVIRONMENT%` and `%COMMAND%` lines are the only lines that are absolutely required in this script and they must appear in this order. These two variables are assigned by Job Control and are not configurable by the user. You might need to add commands to the file after the environment is set up.

The remaining component of this script is the two-line section that sets the environment variable `SCHRODINGER_BATCHID` to the actual batch ID assigned to this job. The batch ID is usually provided by the queueing system in a special environment variable such as the `LSB_JOBID` environment variable used by LSF. The `jmonitor` program checks for the `SCHRODINGER_BATCHID` environment variable and saves the batch ID in the job record, where the user can look it up.

If you want to run parallel Jaguar jobs, which use MPI for parallel execution, you should ensure that the path to `mpirun` (for Linux) or `poe` (for IBM) is prepended to the `PATH` environment variable, by adding the following line after the setting of the batch ID:

`PATH=`*relevant-path*`:$PATH`

3.3 Configuring Support for Grid Computing

From the point of view of Job Control, a grid computing server functions much like a queueing system. Once you have installed the software in the appropriate location for the grid server (see "Preparing for Grid Computing" in the *Installation Guide* for information), you need to set up entries in the hosts file, as described below. This information is also in the *Installation Guide*.

Schrödinger supports two grid servers: the Platform Computing LSF Desktop server, and the United Devices GridMP server. Both of these servers can be used to send job to Windows hosts; the GridMP server can also be used to send jobs to Linux hosts.

There is one important difference from regular queues: Job Control cannot monitor jobs on a grid, and therefore has no information on the progress of the jobs that the grid server sends out to the execution hosts. This information must be obtained from the grid server itself.

3.3.1 LSF Desktop Server

To run jobs on Windows hosts using Platform Computing's LSF Desktop grid server (formerly known as Active Cluster), you must add one or more entries to your hosts file with the following lines:

```
# LSF Desktop (ActiveCluster) server
name:     ac
host:     localhost
platform: WIN32-x86
queue:    AC
qargs:    -W 6:00
```

As for other entries, the `name` setting defines the name of the entry, which is used to direct jobs to the grid. The comment is, of course, also arbitrary.

LSF Desktop is simply a special type of LSF queue, so its configuration is just like any other queue configuration. You must specify AC for the `queue` setting, but you can use the `qargs` setting to specify arguments to be passed to the LSF Desktop queuing system. You can include multiple entries with different `qargs` settings.

3.3.2 GridMP Server

To run jobs on Windows or Linux hosts using the United Devices GridMP server, you must add one or more entries to your hosts file with the following lines:

```
# GridMP server
name:     gridmp
mpconfig: uduserconf
```

As for other entries, the `name` setting defines the name of the entry, which is used to direct jobs to the grid. The comment is, of course, also arbitrary.

The `mpconfig` setting specifies the name of a configuration file containing the contact information for the grid server. The launch scripts look in the current directory and the user's home directory for a file with this name when they need to contact the server. If you specify it using an absolute path name, only that location is checked.

You can include the contact information for the grid server directly in the hosts file, by adding the following settings:

`mpserver:` *grid-server-URL*
`mpfiler:` *file-server-URL*
`mpuser:` *account-name*
`mppassword:` *account-password*

If you also add an `mpconfig` setting, the values in the configuration file override the values in the hosts file.

Files on the UD grid (data files and Windows program modules) all live on the central server and are sent to individual hosts on request. There is no communications between hosts on the grid, and no sharing of files.

3.4 Batch Queue Support on Linux Clusters

This section provides information on the special requirements for running jobs on Linux clusters, using a queuing system. There are many ways of configuring clusters, so for the purposes of this section, one fairly common scenario is used here, as follows:

- The cluster consists of a manager node and a set of compute nodes.
- The compute nodes are on a private network and the management node is on this network and is also accessible externally.
- The manager node is the queue host: that is, the manager node is the only node that can run the queue submission commands.
- Users run Maestro on an external workstation and, from Maestro or a terminal window, submit computational jobs to the queue. This workstation serves as the submission host, and the user's home directory is assumed to be mounted on this host.

The most important requirements are those for how the manager node and the compute nodes communicate between each other and with the outside world. This is where trouble is most likely to arise. The requirements can be stated as follows:

1. Schrodinger software installations must be available to all hosts: the workstation, the manager node, and the compute nodes. These installations must contain the same software versions, but they could be in separate physical installations.
2. The workstation and the compute nodes must be able to open socket connections to the FlexLM license server.
3. Compute nodes must be able to open socket connections to each other and to the external workstation.
4. Passwordless ssh has to be enabled:
    a. from the workstation to the manager node
    b. from the compute nodes to the manager node
5. One of the following three conditions has to be met:
    a. Passwordless ssh has to be enabled from the compute nodes to the workstation.
    b. The directory the user launches the job from and the directory containing the user's job database have to be mounted on the compute nodes.
    c. The directory the user launches the job from and the directory containing the user's job database have to be mounted on the manager node.
6. Unless 5a is met, the firewall that prevents compute nodes from carrying out `ssh` commands to the workstation has to be configured to reject, rather than drop, such traffic.

For parallel Jaguar, the following are required in addition:

- The user's home directory has to be mounted on the compute nodes.
- Each user's home directory has to contain a `.rhosts` file listing all the compute nodes.
- Passwordless `ssh` has to be enabled between compute nodes.

3.5 Additional Information on Queueing Systems

For additional information about PBS, SGE, and LSF, see the following web sites:

PBS: http://www.openpbs.org

http://www-unix.mcs.anl.gov/openpbs

SGE: http://gridengine.sunsource.net

LSF: http://www.platform.com/products/wm/LSF

Please see the notice regarding third-party programs and third-party web sites on the copyright page at the front of this document.

Running Jobs

Jobs can be submitted to a designated host from Maestro or from the command line. The details of setting up the data in Maestro and of submitting jobs from the command line for a given product are described in the documentation for the product. However, there are some common elements that relate to the job submission and execution, and the incorporation of results into a Maestro project. These common elements are described in two sections of this chapter.

When you run a Schrödinger job on a particular host, Job Control determines where to find the hosts file, which version of the software to use, which environment variables need to be passsed to the host, where the scratch directory is located, how to make the input files available to the job, how to retrieve the output files, and how to incorporate them into a Maestro project. An overview of this process is provided in the first section of this chapter. The remaining sections provide details on how Job Control obtains information and sets up the runtime environment for the job.

4.1 The Job Life Cycle

To understand how information is obtained and passed on by Job Control, it it useful to have an understanding of how Schrödinger jobs are run. The "life cycle" of a job can be summarized as follows.

1. A top-level script is run locally (on the *submission host*). This script parses command-line arguments relating to Job Control, sets some environment variables, and locates a software installation that is compatible with the submission host and a software installation that is compatible with the host on which the program will actually be run (the *execution host*).
2. The top-level script then runs the startup script for the program locally. This script parses the command arguments for the program, and assembles information on input and output files for the program.
3. The startup script then runs a job launch script locally, called `jlaunch`. This script creates a record in the job database and populates it with initial values.
4. The job launch script then runs a script to start the actual program on the execution host. This script is called `jmonitor`. It copies input files to the execution host, runs the program, and copies output files back to the submission host.

5. The job launch script takes care of cleanup and incorporation into a Maestro project, if required, and updates the job record with the final status.

If you submit a job to a batch queue, there are some extra tasks that need to be performed. The life cycle of a batch job is as follows.

1. A top-level script is run locally (on the *submission host*). This script parses command-line arguments relating to Job Control, sets some environment variables, and locates a software installation on the submission host and a software installation on the host that manages the batch queue. (the *queue manager*). This host could be one of a few designated nodes on a cluster, for example.
2. The top-level script then runs the startup script for the program locally. This script parses the command arguments for the program, and assembles information on input and output files for the program.
3. The startup script then runs a job launch script locally, called `jlaunch`. This script creates a record in the job database and populates it with initial values. It also creates a batch script for the job and submits this script to the queue.
4. When the queued job is started on an execution host (a cluster node, for example), it runs `jmonitor`. Before performing other tasks, `jmonitor` looks for a compatible software installation on the execution host. It then copies input files to the execution host, runs the program, and copies output files back to the submission host.
5. The job launch script takes care of cleanup and incorporation into a Maestro project, if required, and updates the job record with the final status.

For jobs that are distributed over multiple processors, there is usually a script that manages the distribution of subjobs to the individual processors. Each subjob is then executed by `jmonitor`.

4.2 Running Jobs From Maestro

For many jobs, Maestro opens the Start dialog box so that you can make job settings and select options for handling of the job, including product-dependent options. The controls available in the Start dialog box vary according to the job being run. (Some products and solutions have separate locations in which job settings are made.)

When you start Maestro, the settings in the `schrodinger.hosts` file are used to determine the available options in the Start dialog box and other job setting controls. Maestro searches the following directories for a `schrodinger.hosts` file, in the order given, and uses the first one that it finds.

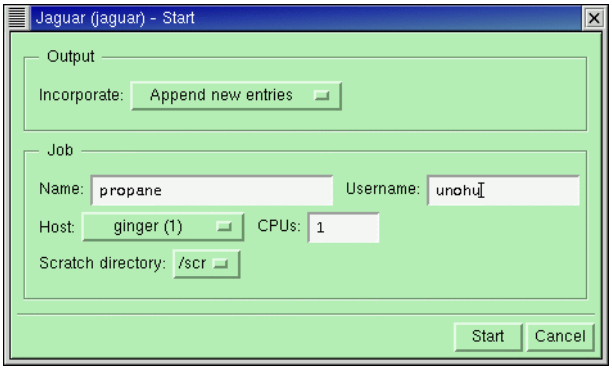

Figure 4.1. The** Start **dialog box.

- The directory in which you started Maestro
- `$HOME/.schrodinger`
- `$SCHRODINGER`

You can always determine which `schrodinger.hosts` file is being used by entering the following command in the directory in which you started Maestro:

`$SCHRODINGER/`*program* `-HOSTS`

where *program* is any Schrödinger executable. The options in the Start dialog box are passed on to Job Control.

The Start dialog box can have two sections, which are described below. An example of a Start dialog box is shown in Figure 4.1.

Output section

- Incorporate—Choose whether the new entries are appended to the project, replace the existing entries, or are not incorporated into the project at all. See Section 4.8 on page 33 for more information on incorporating output.

Job section

- Name—Enter a name for the computational job. This name is used as a prefix for all files created during the calculation. A default name is assigned based on the job being run.

  Note: Maestro does *not* automatically assign new names to jobs or files. If files of the same name exist, a warning is displayed before any files are overwritten.

- Host—Choose a host on which to run the job. This option menu displays all the hosts defined in the hosts file, with the number of processors on the host in parentheses.

- Username—Enter your user name, if it is required for running the job on remote hosts. The default value is the user name of the user who started Maestro, unless there is a `username` setting for this host in the hosts file. If the user name is not correct for the selected host, you can change it in this text box.
- CPUs—Specify the number of processors to use to run the job.
- Scratch directory—Choose a directory to store temporary files.

Once you have finished setting these options, you can click Start to start the job.

4.3 Running Jobs From the Command Line

For most purposes, you can start jobs from Maestro. If you want to run jobs from the command line or from a script, the Job Control facility recognizes a number of command-line options that can be used to control the behavior of the job. These options are summarized in Table 4.1.

Table 4.1. Command options.

Option	Description
`-DISP` *policy*	Set the project incorporation policy for this job: `ignore`, `append`, or `replace`. Requires `-PROJ`, and overrides any user setting of `SCHRODINGER_JOB_DISPOSITION`.
`-HOST` *host*	Run a job on the specified host or submit a job to the specified batch queue. *host* is the value of a `name` entry in the hosts file or the actual address of a host. To specify multiple hosts, supply a blank-separated list in quotes. Default: run on the local host.
`-PROJ` *projectname*	Assign the job to a Maestro project. Overrides any user setting of `SCHRODINGER_PROJECT`
`-TMPDIR` *directory*	Specify the scratch directory for the job. The job directory is created as a subdirectory of the scratch directory. Overrides any user setting of `SCHRODINGER_TMPDIR`
`-USER` *username*	Specify the user name to be used for remote jobs. Must be used with `-HOST`. Default: use the same user name as on the submission host.

In addition to these options, the startup scripts for some programs support several other options. These options are summarized in Table 4.2. You should check which options are supported by entering the command.

```
$SCHRODINGER/program -HELP
```

Command-line options always take precedence over the corresponding environment variable. Some of the options from Table 4.1 and Table 4.2 are described in more detail below.

Table 4.2. Options supported by some programs

Option	Description
`-INTERVAL` *time*	Interval in seconds between updates of the output (usually the log file). The output is copied back to the submission directory at the specified time interval.
`-LOCAL`	Write temporary files in the submission directory instead of the scratch directory. Input and output files are not copied. Default: write temporary files in the scratch directory.
`-NICE`	Run the job at reduced priority.
`-SAVE`	Do not remove temporary files when the job finishes.
`-WAIT`	Wait for the job to finish before executing another command. In a terminal window, this means that the command prompt is not displayed until the job finishes. In a script, it means that the next command is not executed until the job finishes. Default: return control to the shell immediately.

You can also obtain information for each program about the hosts you can use, and which version of the program will be run. These options are listed in Table 4.3 .

Table 4.3. Information options.

Option	Description
`-DEBUG` `-DDEBUG`	Show the details of operation of the toplevel script. Show the verbose details of operation of the toplevel script.
`-ENTRY`	Show the section of the `schrodinger.hosts` file that will be used for this job.
`-HELP`	Display command syntax for the application.
`-HOSTS`	List the hosts that are available for calculations.
`-LIST [`*version-options*`]`	List the available versions of the program that can be run on the host specified by `-HOST`. If no host is specified, the local host is used. If v*ersion-options* is `-ALL`, list all available versions, even if not compatible with the specified host.
`-WHICH [`*version-options*`]`	Show which version of the program and of the `mmshare` library would be used for the given version options.
`-WHY [`*version-options*`]`	Gives information about why the specified version was selected.

The allowed values of *version-options* for the information options are listed in Table 4.4.

Table 4.4. Version options.

Option	Description
`-ARCH` *platform*	Platform code, e.g., `Linux-x86`, `IRIX-mips4`.
`-COMPAT` *executable*	Select the version that is compatible with the specified executable, for which the full path must be given.
`-REL` *version*	Release version number: `v42`, `v4.2`, `42`, `v42062`, `41059`, `v4.1.049` are all acceptable forms.
`-VER` *pattern*	Pattern to match in the path to the executable.

4.3.1 The HOST Option

Jobs can be submitted to a remote host using the `-HOST` option to specify the remote host name. This name must be the value of a `name` setting in the hosts file (a "host entry name"), or the actual address of a host. For example:

```
$SCHRODINGER/bmin -HOST host jobname
```

For programs that can run a single job in parallel or distribute several jobs over a number of processors, the `-HOST` option can be used to specify the list of hosts to be used. The host list is a list of host entry names, separated by spaces. The list must be enclosed in quotes if there is more than one host specified.

Each host entry name can also specify a processor count, using the syntax:

host:*processors*

An example of a host list specification is:

```
-HOST "florence:2 glinda"
```

The first host in the list is the main host for the job, that is, the host on which the master process for the parallel or distributed job executes. However, some distributed jobs specify a separate host for the master process, and some jobs execute the master process locally and submit subjobs remotely.

If you specify a host address rather than a host entry name, the settings for `localhost` in the hosts file are used for the job. You must ensure that there is a software installation on the specified host.

4.3.2 The WAIT option

All jobs are run in the background automatically, dissociated from the terminal session or application from which they were launched. As a result, the job continues to run even if you

quit Maestro or the terminal session from which you launched the job. For command-line jobs, this means that the UNIX command prompt is displayed immediately, without waiting for the job to finish. This behavior is not always desirable, especially if you want to run the job in a script, in which some subsequent action can be taken only after the job finishes. The command-line option `-WAIT` can be used to prevent the shell from continuing to the next command until after the job finishes. For example:

```
$SCHRODINGER/bmin -WAIT job_name
```

Even with this option, however, the calculation is placed in the background, so pressing CTRL+C or CTRL+Z does not affect it. This option applies to jobs submitted to a batch queue as well as jobs that are run directly.

4.3.3 The LOCAL Option

For some applications, you can request temporary files to be written to the submission directory (the directory from which you submitted the job), instead of to the job directory, which is a subdirectory of the scratch directory. This is done with the `-LOCAL` option. For example,

```
$SCHRODINGER/bmin -LOCAL jobname
```

Not every application supports this option. When it is supported, it may be used for both local and remote jobs. If you run a remote job with the `-LOCAL` option, it is important to make sure that the submission directory is accessible on the remote machine. This option also suppresses the copying of input and output files.

4.3.4 Location of the Hosts File

When you start a job from the command line, one of the first tasks of Job Control is to locate a hosts file. The hosts file used for a given job is the first one found in the following list:

1. The file specified by the environment variable `SCHRODINGER_HOSTS`
2. The `schrodinger.hosts` file in the current directory
3. The `schrodinger.hosts` file in `$HOME/.schrodinger`
4. The `schrodinger.hosts` file in `$SCHRODINGER`

For MacroModel jobs the following locations are searched before the list given above:

1. A file specified by the command line argument `-HOSTFILE`.
2. The file *jobname*`.hst` in the startup directory on the submission host, where *jobname* is the stem of the command file name for the current calculation (e.g., if the command file were called `cal_en1.com`, this file would be called `cal_en1.hst`).

The information in the hosts file is then used to make default settings for the job, such as the location of the scratch directory.

4.4 Location of the Scratch Directory

Most jobs now run in a scratch directory by default, rather than in the directory from which the job was started (the *submission* directory). When a job runs in a scratch directory, a subdirectory is created in it for the job, named *tmpdir*/*username*/*unique_name*. Here, *tmpdir* is the path to the scratch directory. The job name is usually used for *unique_name*, but it can also have a sequence number appended to it in order to make the directory name unique. This subdirectory is called the *job directory*. Input files are copied to the job directory, temporary files, log files, and output files are created in the job directory, and the output and log files are copied back to the submission directory when the job finishes.

If you submit a job from Maestro, you may be able to choose the scratch directory in the Start dialog box. Maestro reads all the `tmpdir` settings from the hosts file, and presents these in the Scratch directory option menu.

For jobs submitted from the command line, there are a number of ways to specify the *tmpdir* directory. Job Control uses the first specification found from the following list:

- The directory given on the command line with the `-TMPDIR` option. For example,

  ```
  $SCHRODINGER/bmin -TMPDIR /scr/mmod_tmp job_name
  ```

- The directory specified by the `SCHRODINGER_TMPDIR` environment variable, if this is set on the submission host.
- The directory specified by the first `tmpdir` setting for the host entry in the hosts file.

For jobs run on a remote host, the following locations are considered after the ones listed above if a scratch directory is not defined:

- The directory specified by the environment variable `SCHRODINGER_TMPDIR` on the remote machine.
- The directory specified by the environment variable `TMPDIR` on the remote machine.

If no other specification for *tmpdir* is found, the directory `$HOME/.schrodinger/tmp` is used. In this case, the *username* is not used to form the job directory name, since it would be redundant. Temporary files can occupy a large amount of disk space, so you are strongly advised to ensure that a suitable directory is defined on every host, preferably in the hosts file.

In all cases, *tmpdir* is created if it does not exist. If the file system on which *tmpdir* is to be created does not exist, the job fails.

When the job finishes, the job directory is automatically removed, if the following conditions are met:

- The output files were all successfully copied back to the job startup directory.
- The directory did not exist before the job started.
- The `-SAVE` option was not used in the job submission.

Some programs allow you to force the storage of temporary files in the submission directory, by using the `-LOCAL` option to the command for running the program—see Section 4.3.3 on page 29.

4.5 Software Version Selection

When a job is started, the Job Control facility searches for compatible software versions that are available on the execution host. Compatibility here means that the platform matches the execution host. The first location on the following list that contains a compatible software version is used:

1. Software in the directory specified by `$SCHRODINGER` on the submission host.
2. Software in the directory specified by the `schrodinger` settings in the entry for the host and the `localhost` entry in the hosts file.

   The hosts file used is the one on the submission host. These `schrodinger` settings are compiled into a list, which can be displayed by entering the command for the program with the `-LIST` option in a terminal window, for example:

   `$SCHRODINGER/`*program* `-LIST`

If there is more than one compatible version, the most recent is used.

Thus, all the information on the software versions must be available on the submission host. You must therefore ensure that the version you want to use is specified in the hosts file. You can specify different versions for a given host entry by including multiple `schrodinger` definitions in the hosts file on the submission host. Checking of the availability of the software is done on the execution host, however.

If you submit a job to a batch queue, an additional check is done when the job starts, using the information obtained on the submission host, which is passed on by Job Control. This is done because when the job is launched from the submission host, Job Control can only check the availability of software on the queue manager.

It is possible that an older version of the software could be used to run a job if, for example, the software on the submission host was older that the software on the execution host. You should therefore ensure that the software installation on the submission host is up-to-date.

If you are running a job from the command line, you can select a version, a release, and an architecture by specifying command-line options, as detailed in Section 4.3 on page 26.

4.6 Environment Variables

Environment variables that can be used on the execution host can come from the following sources:

- Environment variables set in the shell startup script (`.bashrc`, `.cshrc`, and so on)
- Environment variables set on the submission host when the job is launched.
- Environment variables set in the hosts file for the execution host or `localhost`.

The environment variables are passed to the job in the following order of precedence:

1. Settings in the host entry.
2. Settings in the user environment on the submission host.
3. Settings in the user environment on the execution host. These are used as a fallback if those environment variables were not defined and exported with the job.

4.7 Input and Output Files

When you run a job from the command line you can specify input files in either of two ways: using absolute paths or relative paths. The way Job Control treats these differs. The configuration considerations for output files and for updates of the Job Control database are similar to those for input files specified using absolute paths. The basic rules are as follows:

- **Files specified using absolute paths.** These are paths that begin with a forward slash (/) such as `/home/unohu/prime/1mcp.mae`. Job Control first looks for such a file on the execution host exactly as typed. For instance, it would look for a file named `/home/unohu/prime/1mcp.mae` on the execution host. If it finds the file, it does nothing more, and the file on the execution host is used for input. If it does not find the file, it looks for a file of that name on the submission host, and copies it to the job directory of the execution host.

  If you specify an absolute path for a file on a remote host, you should check that this is the file you intended to use. There is no guarantee that the file that appears at the specified location on the execution host is the same as or has the same contents as one that appears at the specified location on the submission host; nor is there a requirement that a file of this name be accessible on the submission host.

  Specifying file names with absolute paths is useful if you have large files that you can copy to the desired location before submitting the job. For example, you might want to

run several Glide docking jobs with the same grid. You could copy the grid files to the desired location, then specify them with an absolute path. Copying large files prior to job submission can help reduce network traffic, especially on clusters.

- **Files specified using relative paths with ..** These files are copied from their locations on the submission host to the job directory on the execution host. For example, the file `../prime/1mcp.mae` is copied to *jobdir*`/1mcp.mae`.
- **Files in the submission directory or its subdirectories.** These files are copied with their relative path to the job directory on the execution host. Any subdirectories are created in the job directory. For example, `1mcp.mae` is copied to *jobdir*`/1mcp.mae`, and `prime/1mcp.mae` is copied to *jobdir*`/prime/1mcp.mae`.

Output files are copied back from the job directory to the submission directory.

4.8 Incorporation of Job Output

When Maestro project entries are used as input for jobs, the structure and property output can be incorporated into the project. You can specify how you want the results to be incorporated in the Start dialog box (see Section 4.2 on page 24). To open the Start dialog box, click the Start button in the panel used to set up the job. The three incorporation options and their effects are described below.

- Append new entries—New entries are added to the end of the entry list in the project. The names for the new entries have `-inc`*n* appended, where *n* is a number used to ensure uniqueness of the name.
- Replace existing entries—Incoming entries replace pre-existing entries of the same name.
- Do not incorporate—Output files are copied back to the submission directory but structure and property data are not incorporated into the Project Table. You can import the results later using the Import panel. See Chapter 3 of the *Maestro User Manual* for more information on importing structures.

You can also specify a project and incorporation mode when you run jobs from the command line, by using the `-PROJ` and `-DISP` options—see Table 4.1 on page 26.

Incorporation of job output into the project takes place only from monitoring mode in Maestro. If you are monitoring a job launched from the current project, and that job finishes during the monitoring session, incorporation is immediate. If you are not in monitoring mode when the job finishes, you must select and then monitor the job before the results are incorporated. This applies also to jobs run from the command line and associated with a project: you must monitor the job in the Monitor panel for incorporation to take place.

When a job is incorporated, the first structure in the job output is displayed in the Workspace. Entries are incorporated as an entry group, and the entry list is scrolled to the first of these entries.

Incorporation can be undone. Undoing incorporation removes the new entries from the project and restores the Workspace to the state it was in just before monitoring mode was entered.

Computational programs normally propagate entry names (as well as other properties) from their input into their output structures. Where the relationship is one-to-many, as in a conformational search, consecutive structures in the output file have identical entry names. If the job is run with the "append" incorporation mode, the names are identical to that of the input entry.

Because project entries must have unique names, the project facility uses automatic rules for creating unique entry names when the output entries are incorporated. When an entry is incorporated, the suffix `-inc`*n* is added to the entry name. The value of *n* starts at 1 and is incremented for each incorporation event. For instance, if the input entry was `ligand1` and the job output contained a single entry, also named `ligand1`, the output entry would be given the name `ligand1-inc1`.

Multiple consecutive entries bearing the same entry name incorporated from a single file are distinguished using a "dot suffix." Thus, if the output of the above job contained multiple consecutive entries, all named `ligand1`, then, on import, they would be renamed `ligand1-imp1.1`, `ligand1-imp1.2`, etc.

See Appendix C of the *Maestro User Manual* for information on the suffixes used in entry naming.

Managing Jobs

The Job Control facility provides tools for monitoring, and controlling the jobs that it runs. These tools have both a graphical user interface in the Maestro Monitor panel and a command-line interface in the `jobcontrol` command. The interface gets information from a job database that is set up for each user, and uses this information to perform various tasks, like providing job status and killing jobs.

You can monitor jobs that are run locally, run remotely, or run in a batch queue. Jobs submitted to a grid computing manager cannot be monitored, because Job Control does not have information from the grid manager.

The first section in this chapter describes the job database. Subsequent sections describe the Monitor panel and the `jobcontrol` command.

5.1 The Job Database

Information about each job is kept in the user's job database. The database contains a record for each job. You can determine which jobs are in the database by using the Maestro Monitor panel or the command-line `jobcontrol` utility.

By default, this database is kept in the directory `$HOME/.schrodinger/.jobdb`. If you want to change the location, you can set the environment variable `SCHRODINGER_JOBDB` to the desired location. You must ensure that this database can be read and written by a job running on any host, either by making the directory directly available on the host, or by ensuring that the host has access to a host on which it is available by passwordless `rsh` or `ssh`. If a job does not have access to the database, it will not be updated.

5.1.1 The Job Record

Each job has a job record in the job database. The job record is a list of fields, one on each line, each consisting of a field name and its value. Many of these fields contain information that is only useful to the job control system, but a number may also be useful to users. Some of the latter are listed in Table 5.1

To list the complete database record for a job, enter the command:

```
$SCHRODINGER/jobcontrol -dump jobid
```

Table 5.1. Fields used in the job record.

Field Name	Meaning
BackendPid	The PID for the program carrying out the calculation
Command	The command used to start up the actual calculation
Dir	The submission and output directory (on Host)
Envs	Environment variable settings for the job
Errors	Error messages from the job control system
ExitStatus	The reason the calculation stopped
Host	The machine from which job was launched
InputFiles	Files copied from the submission directory to the job directory at startup
JobDir	The directory in which the job is run
JobHost	The machine on which the job is run
JobId	The job's JobId
JobPid	The PID for the job's `jmonitor` process, or the job's `jlaunch` process before jmonitor has started.
JobUser	The user name under which the job is run
LaunchTime	The time at which the job was submitted
LogFiles	Monitoring files that grow throughout the job
MonitorFiles	Files copied from the job directory to the output directory during monitoring
MonitorInterval	The interval in seconds between monitoring updates (0 if off)
Name	The job name
OutputFiles	Files copied from the job directory to the output directory at exit
ParentJobId	The JobId of the parent job, if this is a subjob
Processors	The number of processors used for a parallel or distributed job
Program	The program name
Project	The project name
StartTime	The time at which the calculation started running
Status	The current job status
StatusTime	The time at which Status was last updated
StopTime	The time at which the calculation stopped

Table 5.1. Fields used in the job record. (Continued)

Field Name	Meaning
StructureMonitorFile	The name of the monitoring file holding the molecular structure
StructureOutputFile	The name of the file holding the final molecular structure
User	The user name under which job was launched

5.1.2 Job Status

Every job proceeds through a series of well-defined stages (see Section 4.1 on page 23). The current stage of a job is displayed in the Status column of the Monitor panel. The Status of a completed job indicates the conditions under which it stopped. The job status descriptors that can appear in the Monitor panel are listed in Table 5.2. When program execution finishes, an exit status is returned, and is displayed with the completed status and the incorporated status in the Monitor panel. The exit status descriptors are listed in Table 5.3.

Table 5.2. Job status descriptors in the Monitor *panel.*

Status	Meaning
launched	The job was submitted and assigned a JobId.
submitted	The job is in a batch queue, waiting to be scheduled.
started	The environment for the job is being set up.
running	The program is running.
paused	The program has temporarily been suspended.
exited	The program has stopped and the job is being cleaned up.
completed	The job has been cleaned up. This descriptor is followed by a colon and the exit status (see Table 5.3).
incorporated	The job results have been incorporated into a Maestro project.
stranded	Job Control could not retrieve results or clean up the job.
unreachable	Job Control could not connect to the host running the job.

Table 5.3. Exit status descriptors in the Monitor *panel.*

Status	Meaning
finished	The program finished successfully.
stopped	The program was stopped at an appropriate point at the user's request.
killed	The job was killed by someone (not necessarily the user).
died	The program failed during execution.
fizzled	The job failed before the program could be run.

5.2 Managing Jobs From Maestro

You can manage, or control, jobs from the Monitor panel or from the command line. The common tasks can be performed from the Monitor panel, but the full range of job control tasks is only available from the command line.

When a computational job is started, Maestro immediately goes into monitoring mode. The Monitor panel opens, and the log file for the job is displayed in the text area at the top of the panel. The display is updated as the log file changes. For some kinds of jobs, such as MacroModel and Jaguar geometry optimizations, the Workspace is updated with each new geometry as it is generated.

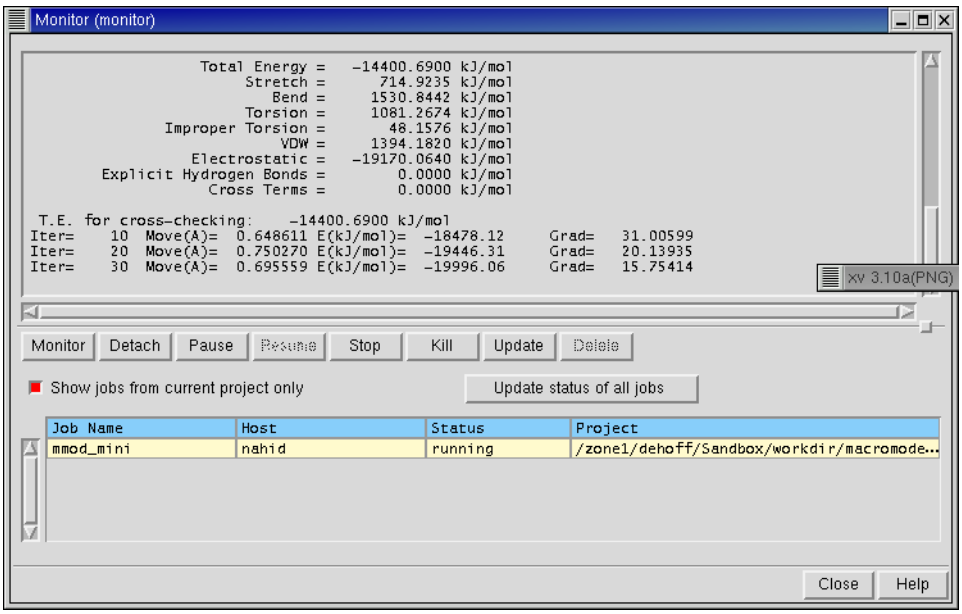

Figure 5.1. The* Monitor *panel.

Note: If the job directory (the directory containing the input and output files for the job) is not mounted on the execution host, the job cannot be monitored in the Monitor panel or by `jobcontrol -view jobid` (issued at the command line).

Most Maestro operations stop the monitoring of a job. You can also exit monitoring mode by clicking Detach on the Monitor panel. You can resume or begin monitoring a job at any time by selecting it from the list of jobs in the Monitor panel.

You can monitor any job that uses Schrödinger software from the Monitor panel. The job control facility keeps a database of all your jobs on all hosts, regardless of whether they were initiated from Maestro or from the command line.

If you want to monitor a job and the Monitor panel is not open, choose Monitor Jobs from the Applications menu.

The Monitor panel allows jobs to be paused, resumed, or terminated using the Pause, Resume, Stop, and Kill buttons. Clicking Kill terminates the selected computational job immediately. The Stop button only affects MacroModel and Jaguar jobs, and is ignored by other jobs. When a job is stopped, the program saves results from the current stage of execution and cleans up before exiting. Pause and Resume can be used for all jobs.

5.3 Managing Jobs From the Command Line

The job control utility allows you to perform a number of job control tasks from the command line. The syntax for job control utility commands is:

```
$SCHRODINGER/jobcontrol command query
```

where *command* is the command for the action you want to perform, and *query* defines the scope of the action performed by the command. Valid commands are listed in Table 5.4:

The optional *query* consists of one or more JobIds, job names, status codes or queries, or the keywords `all` or `active`. The default *query* is all active jobs (jobs that are not finished), and is equivalent to using the keyword `active`. The keyword `all` means all jobs in your job database. The JobId is a unique identifier consisting of the name of the submission host, a sequence number, and a hexadecimal timestamp, e.g., `mirabelle-0-a1b2c3d4`. The job record fields and their meanings are listed in Table 5.1 on page 36.

The following examples illustrate different values of *query*.

- To list all active jobs, showing their JobIds, job names, current status and the machine on which each is running, enter:

  ```
  $SCHRODINGER/jobcontrol -list
  ```

Table 5.4. Jobcontrol commands.

Command	Action
`-abort`	emergency stop, abandon output files and the job record
`-cancel`	cancel a job that has been launched, but not started
`-delete [-force]`	remove a completed job from the database. `-force` removes any job from the database, whether completed or not, and should only be used when other methods fail.
`-dump`	show the complete job record
`-h[elp]`	describes the job control commands
`-int` *seconds*	interval for checking job status (default: 5 seconds)
`-kill`	terminate the job immediately
`-killnooutput`	terminate the job immediately and discard the output
`-list`	list the JobId, job name and status
`-monitor` *n*	ask for monitoring files to be sent every *n* seconds
`-pause`	suspend the job temporarily
`-ping`	verify that a running job responds to job control messages
`-recover`	try to retrieve results from a stranded job
`-refresh`	update the status of the job
`-resume`	continue running a paused job
`-show`	show more details of the job status
`-stop`	ask the job to stop itself as soon as possible (Jaguar, MacroModel only)
`-update`	ask for an update of the job results. (MacroModel only)
`-v[ersion]`	display program version and exit
`-view [-l` *log*`]`	view the job's log files in real time
`-wait`	wait for the job to finish before returning to the command prompt

- To list all the jobs in your job database that finished successfully, enter:

```
$SCHRODINGER/jobcontrol -list finished
```

- To list just the job whose JobId is `serv01-0-a1b2c3d4`, enter:

```
$SCHRODINGER/jobcontrol -list serv01-0-a1b2c3d4
```

- To list all jobs in your database with the job name `myjob`, enter:

```
$SCHRODINGER/jobcontrol -list myjob
```

- To list all jobs in your database, enter:

```
$SCHRODINGER/jobcontrol -list all
```

You can also use the wildcard character '?' to match a single unspecified character, or '*' to match zero or more unspecified characters. If you use either of these characters, you must protect them to ensure that they are interpreted by the `jobcontrol` script and not the UNIX shell. For example, you could enter either of the following commands to list all jobs whose job names start with `docklig`

```
$SCHRODINGER/jobcontrol -list docklig\*
$SCHRODINGER/jobcontrol -list 'docklig*'
```

5.3.1 General Job Control Queries

More general queries than those given above are also possible. Formally, a query consists of one or more *conditions*, optionally separated by the Boolean operators AND, OR or NOT. If the operators are omitted, OR is assumed. A condition takes the form <field><op><value>, where <field> is one of the field names in the job record and <op> is one of the following:

=	equals
!=	is not equal to
=~	matches
!~	does not match

"Equals" means an exact match; "matches" means that <field> matches <value>, treated as a regular expression.

The field names are listed in Table 5.1 on page 36. The fields most likely to be useful for queries are Name, Program, Host, Dir, JobHost, JobDir, Project, Status, and ExitStatus. The case of the field names is ignored, but the case of the <value> is significant. So, `Program=Jaguar` is the same as `program=Jaguar`, but `program=jaguar` would fail. Parentheses can be used to group conditions, but these must also be protected if used on the Unix command line. For example, to list all QSite jobs in your database that either died or were killed, enter the command:

```
$SCHRODINGER/jobcontrol -list program=QSite AND
    \( died OR killed \)
```

5.3.2 Recovering Stranded Jobs

If a job is stranded, it might still be possible to re-establish contact with the job and incorporate the results. This can be done with the command:

```
$SCHRODINGER/jobcontrol -recover jobid
```

If this command fails, however, the job is cleaned up and the results are no longer available.

If you cannot recover a job you can remove the job record with the command:

```
$SCHRODINGER/jobcontrol -delete -force jobid
```

This command should only be used as a last resort, because it removes all job control information for the given job.

5.3.3 Purging the Job Database

Normally, when running jobs from Maestro's project facility, the database record for a job is cleared once the job is incorporated into a Maestro project. Not all jobs can be incorporated, however; for this and other reasons job records might not get deleted automatically and the job database directory can become quite large over time. The `jobcontrol` utility can be used to purge the job database of records for completed jobs. For example, the following command purges the entire database:

```
$SCHRODINGER/jobcontrol -delete all
```

This command only affects completed jobs: running jobs cannot be deleted unless `-force` is supplied after `-delete`. However, you should be careful not to delete completed jobs whose output you still intend to incorporate into a Maestro project.

The job database is automatically checked for jobs that have finished whenever you start a new job. If the job is older than a threshold time, it is deleted from the database. You can set this threshold time in the environment variable `SCHRODINGER_JOBDB_CLEANUP`. The minimum time is 1 second, and the default time is 1 week. The default unit is seconds, but you can specify a time in minutes, hours, or days by appending m, h, or d to the value, for example, 7d, 168h, or 5m.

Getting Help

Schrödinger software is distributed with documentation in PDF format. If the documentation is not installed in `$SCHRODINGER/docs` on a computer that you have access to, you should install it or ask your system administrator to install it.

For help installing and setting up licenses for Schrödinger software and installing documentation, see the *Installation Guide*. For information on a particular product, see the documentation for that product. For information on using Maestro and its panels, see the Maestro online help or the *Maestro User Manual*. Other information, including manuals and FAQ, is available on the Schrödinger Support Center.

If you have questions that are not answered from any of the above sources, contact Schrödinger using the information below.

E-mail:	help@schrodinger.com
USPS:	101 SW Main Street, Suite 1300, Portland, OR 97204
Phone:	(503) 299-1150
Fax:	(503) 299-4532
WWW:	http://www.schrodinger.com
FTP:	ftp://ftp.schrodinger.com

Generally, e-mail correspondence is best because you can send machine output, if necessary. When sending e-mail messages, please include the following information, most of which can be obtained by entering `$SCHRODINGER/machid` at a command prompt:

- All relevant user input and machine output
- Schrödinger software purchaser (company, research institution, or individual)
- Primary Schrödinger software user
- Computer platform type
- Operating system with version number
- Schrödinger software version number
- Maestro version number
- mmshare version number

Glossary

entry—(1) A collection of settings in the hosts file that defines a configuration for running jobs on a given host. There can be more than one entry for a given host, with different settings. (2) A structure and its properties in a Maestro project.

execution host —The computer that a job runs on.

hosts file—The file that contains information on available hosts that is used by Job Control. This file is usually named `schrodinger.hosts`.

job directory—The directory on the execution host to which the input files are copied from the submission host and from which the output files back to the submission host when the job is finished. This is usually a subdirectory of the scratch directory that is created by Job Control.

local host—The computer that you are logged on to. Usually this is also the computer you are using for job submission.

output directory—The directory to which output files are copied at the end of the job. This is usually the same as the submission directory.

queue host—A computer that can run the queueing software. This is not necessarily the queue manager: for example, if the queuing software is installed on an NFS-mounted file system, it could be any host that has that file system.

remote host—A computer that is available to you over a network.

remote job submission—Submitting a job to a computer other than the one you are logged on to or that you are running Maestro on.

scratch directory—The directory that is used for temporary files. The files are usually created in a subdirectory of this directory that is unique to the user and the job.

submission directory—The directory from which the job is started. This is the directory from which input files (specified without a path) are copied to the job directory, and to which output files are copied at the end of the job.

submission host—The computer that you submit a job from.

Index

K

L

M

O

P

Q

R

S

T

U

120 West 45th Street
32nd Floor
New York, NY 10036

101 SW Main Street
Suite 1300
Portland, OR 97204

3655 Nobel Drive
Suite 430
San Diego, CA 92122

Dynamostraße 13
68165 Mannheim
Germany

QuatroHouse, Frimley Road
Camberley GU16 7ER
United Kingdom

SCHRÖDINGER.